# Computer Graphics Lab 3 Report

一、实验内容

Bezier 曲线绘制的 de Casteljau 算法

B 样条曲线绘制的 de Boor-Cox 算法

二、测试环境

OS: Microsoft Windows 10 家庭中文版 64Bit

CPU: Intel® Core™ i7-6500U @ 2.50GHz

RAM: 8GB DDR4 2133MHz

Programming Language: Microsoft Visual C++

IDE: Microsoft Visual Studio 2017 Community  Compiler Version: MSVC 14.10.25017

程序使用了 freeglut 3.0 调用 OpenGL 绘图，绘制算法和所需数据结构在 MyMath.h 和 Main.cpp 中，主要框架在 Main.cpp 中。

三、操作说明

本程序一切操作通过鼠标和键盘交互完成，初始界面为空，单击鼠标左键可在当前位置添加控制顶点，在顶点上单击 Backspace 键即可删除该顶点。顶点处按住鼠标左键可提起该顶点，使其随鼠标移动，松开鼠标左键可放下该顶点。每次操作后都会即时画出当前顶点确定的 Bezier 曲线和 B 样条曲线。单击鼠标右键可以选择显示选项。

同一位置或接近位置有多个顶点，删除时按照后加入的先删除规则进行。

四、实现过程

使用如下数据结构储存控制顶点：

```
const float PI = 3.1415926f;
const float FLOAT_EPS = 2e-2f;

struct Vector2f {
	float x, y;

	Vector2f(float x = 0.0, float y = 0.0) : x(x), y(y) {
	}

	Vector2f operator * (const float lambda) {
		return Vector2f(x * lambda, y * lambda);
	}

	Vector2f operator + (const Vector2f & tempVector2f) {
		return Vector2f(x + tempVector2f.x, y + tempVector2f.y);
	}

	Vector2f operator - (const Vector2f & tempVector2f) {
		return Vector2f(x - tempVector2f.x, y - tempVector2f.y);
	}

	float Modulus() {
		return sqrt(x * x + y * y);
	}

	bool Between(const Vector2f & headVector2f, const Vector2f & tailVector2f) {
		if (x > headVector2f.x && x > tailVector2f.x || x < headVector2f.x && x < tailVector2f.x)
			return false;
		if (y > headVector2f.y && y > tailVector2f.y || y < headVector2f.y && y < tailVector2f.y)
			return false;

		if (abs((x - headVector2f.x) * (y - tailVector2f.y) - (y - headVector2f.y) * (x - tailVector2f.x)) > FLOAT_EPS)
			return false;
		return true;
	}
};
```

所有顶点信息通过鼠标动作监测加入顶点列表，所有鼠标动作通过 OpenGL 的 glutMouseFunc()和 glutMotionFunc()函数监控，所有键盘动作通过 OpenGL 的 glutKeyboardFunc()函数监控。每次操作结束后，使用新的顶点列表调用 de Casteljau 和 de Boor-Cox 算法，使用 OpenGL 绘制出一系列点形成目标曲线。

```
void DrawBezier(float lambda) {
	vector<Vector2f> drawVertex(vertexArray);

	for (int i = 1; i < drawVertex.size(); i++)
		for (int j = 0; j < drawVertex.size() - i; j++)
			drawVertex[j] = drawVertex[j] * (1 - lambda) + drawVertex[j + 1] * lambda;

	glVertex3f(drawVertex[0].x, drawVertex[0].y, 0.0);
}

void DrawBspline(int curr, float lambda) {
	vector<Vector2f> drawVertex(vertexArray);

	for (int i = 1; i < BsplineOrder; i++)
		for (int j = curr; j > curr - BsplineOrder + i; j--) {
			float realLambda = (lambda - BsplineT[j]) / (BsplineT[j + BsplineOrder - i] - BsplineT[j]);
			drawVertex[j] = drawVertex[j] * realLambda + drawVertex[j - 1] * (1 - realLambda);
		}

	glVertex3f(drawVertex[curr].x, drawVertex[curr].y, 0.0);
}
```

五、结果示例